Tarea 2

Redes Sociales y Económicas

Daniel Ramos, Jordi Vanrell, Sergi Fornes

17/11/2020

We shall consider again the undirected Facebook friendship network considered in the last handout. The links in this network are contained in the file **facebook_sample_anon.txt**. Download it on your computer and upload it to R as a dataframe. Define an undirected graph with this list of edges.

Primero de todo cargamos el archivo, lo metemos en un data frame y lo convertimos en grado no dirigido.

```r
data <- read.table("./data/facebook_sample_anon.txt")
gf <- graph_from_data_frame(d = data, directed = F)
```

1) It has been observed in many networks an association between "centrality" and "lethality," defined as the fatal disconnection of the network when nodes are removed. Let's study this association on this network.

a) Repeat 1000 times the procedure of removing a random 0.1% of its set of nodes, and compute the average number of connected components of the resulting networks and the average fraction of the network represented by the largest component. Use **set.seed** to make your results reproducible.

```r
set.seed(2020)
vec_con <- c()
vec_frac <- c()
# Miramos cuantos nodos tenemos que quitar
round(0.001 * length(V(gf)))
```

```
## [1] 4
```

```r
# Creamos un bucle en el que vamos quitando al grafo original 4 nodos aleatorios
#y analizamos el grafo resultante
for (i in 1:1000){
  s <- sample(V(gf), size = 4, replace = FALSE)
  n_gf <- delete_vertices(gf, s)
  vec_con[i] <- components(n_gf)$no
  vec_frac[i] <- max(components(n_gf)$csize) / sum(components(n_gf)$csize)
}
mean_con <- mean(vec_con)
mean_frac <- mean(vec_frac)
```

La media del número de componentes conectados cuando quitamos 4 nodos aleatoriamente es de 1.155.

La media de la fracción de la red que representa el componente con más nodos es 0.9999.

---

b) Now, compute the number of connected components and the fraction represented by the largest component of the networks obtained after removing the most central 0.1% of nodes, for the following centrality indices (of course, if the most central 0.1% of nodes for two indices are the same set of nodes, you need not waste your time considering twice the same network): *degree*; *closeness*; *betweenness*; *page.rank*. (**Hint**: It might be convenient to define first a function that removes a given set of nodes of this graph and computes the number of connected components and the fraction represented by the largest component of the resulting network; then you will only need to apply it to the required different sets of most central nodes.) Is it what you expected?

Usando el *degree* como medida de centralidad:

```
# Buscamos los nodos con mayor degree
V(gf)$degree <- degree(gf)
max_degree <- sort(V(gf)$degree, decreasing = TRUE)[1:4]
central_v_degree <- V(gf)[V(gf)$degree %in% max_degree]
# Eliminamos los nodos más centrales
n_gf <- delete_vertices(gf, central_v_degree)
# Analizamos el grafo resultante
degree_con <- components(n_gf)$no
degree_frac <- max(components(n_gf)$csize) / sum(components(n_gf)$csize)
```

Usando el *closeness* como medida de centralidad:

```
# Buscamos los nodos con mayor closeness
V(gf)$closeness <- closeness(gf)
max_closeness <- sort(V(gf)$closeness, decreasing = TRUE)[1:4]
central_v_closeness <- V(gf)[V(gf)$closeness %in% max_closeness]
# Eliminamos los nodos más centrales
n_gf <- delete_vertices(gf, central_v_closeness)
# Analizamos el grafo resultante
closeness_con <- components(n_gf)$no
closeness_frac <- max(components(n_gf)$csize) / sum(components(n_gf)$csize)
```

Usando el *betweenness* como medida de centralidad:

```
# Buscamos los nodos con mayor betweenness
V(gf)$betweenness <- betweenness(gf)
max_betweenness <- sort(V(gf)$betweenness, decreasing = TRUE)[1:4]
central_v_betweenness <- V(gf)[V(gf)$betweenness %in% max_betweenness]
# Eliminamos los nodos más centrales
n_gf <- delete_vertices(gf, central_v_betweenness)
# Analizamos el grafo resultante
betweenness_con <- components(n_gf)$no
betweenness_frac <- max(components(n_gf)$csize) / sum(components(n_gf)$csize)
```

Usando el *page.rank* como medida de centralidad:

```
# Buscamos los nodos con mayor pagerank
V(gf)$pageRank <- page_rank(gf, directed = F)$vector
max_pr <- sort(V(gf)$pageRank, decreasing = TRUE)[1:4]
central_v_pr <- V(gf)[V(gf)$pageRank %in% max_pr]
# Eliminamos los nodos más centrales
n_gf <- delete_vertices(gf, central_v_pr)
```

```r
# Analizamos el grafo resultante
pr_con <- components(n_gf)$no
pr_frac <- max(components(n_gf)$csize) / sum(components(n_gf)$csize)
```

Medida de centralidad	Nº de componentes conectados	Fracción que representa el mayor componente
Random	1.16	0.999941
Degree	41	0.9879
Closeness	12	0.9973
Betweenness	41	0.9879
PageRank	52	0.9839

Como es de esperar, quitar 4 nodos centrales tiene mucho mayor efecto en la conexión del grafo que quitarlos aleatoriamente. Con estos datos, es más letal quitar los nodos usando la medida de PageRank, por lo que en este caso representaría mejor la centralidad de los nodos del grafo.

---

2) Now, consider the same graph as a directed one, and find the hubs and authorities scores. Compare with the page rank score.

```r
gf_d <- graph_from_data_frame(d = data, directed = T)
# Calculamos los valores pagerank, hub y authority de los nodos del grafo dirigido
V(gf_d)$pageRank <- page_rank(gf_d, directed = T)$vector
V(gf_d)$hs <- hub_score(gf_d, weights = NA)$vector
V(gf_d)$as <- authority_score(gf_d, weights = NA)$vector

# Buscamos los 10 nodos con mayor pagerank
dmax_pr <- sort(V(gf_d)$pageRank, decreasing = TRUE)[1:10]
dcentral_v_pr <- V(gf_d)[V(gf_d)$pageRank %in% dmax_pr]

# Buscamos el hub y el authority de estos 10 nodos
V(gf_d)$hs[V(gf_d) %in% dcentral_v_pr]
```

```
##  [1] 6.513024e-07 4.275055e-06 7.147870e-03 0.000000e+00 0.000000e+00
##  [6] 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00 0.000000e+00
```

```r
V(gf_d)$as[V(gf_d) %in% dcentral_v_pr]
```

```
##  [1] 5.998040e-05 3.446861e-05 8.198232e-01 2.473895e-06 1.963741e-06
##  [6] 5.388325e-04 6.731517e-02 8.066440e-09 9.009250e-01 2.972833e-10
```

```r
# Buscamos los 10 nodos con mayor pagerank
dmax_pr <- sort(V(gf_d)$pageRank, decreasing = TRUE)[1:10]
dcentral_v_pr <- V(gf_d)[V(gf_d)$pageRank %in% dmax_pr]
# Buscamos los 10 nodos con mayor hub
dmax_hs <- sort(V(gf_d)$hs, decreasing = TRUE)[1:10]
dcentral_v_hs <- V(gf_d)[V(gf_d)$hs %in% dmax_hs]
# Buscamos los 10 nodos con mayor authority
```

```r
dmax_as <- sort(V(gf_d)$as, decreasing = TRUE)[1:10]
dcentral_v_as <- V(gf_d)[V(gf_d)$as %in% dmax_as]

#cor(pageRank, hs)
#cor(pageRank, as)
#plot(pageRank, hs)
#plot(pageRank, as)


#plot(gf_d, vertex.size=hs*50, main="Hubs")
#plot(gf_d, vertex.size=as*30, main="Authorities")

library(networkR)
# data <- read.table("facebook_sample_anon.txt")
# head(data)
# gf_d <- graph_from_data_frame(d=data, directed = T)
#kmax<-20 # 20 iteraciones
#gf_d_mtx <- as_adjacency_matrix(gf_d)
#op<-hits(gf_d_mtx,kmax)
#op
#str(op$authorities)
#str(op$hubs)
```